



# **Interface Specification**

## **Ver. 2.0**



**REVISION HISTORY**

<b>Revision</b>	<b>Issue Date</b>	<b>Author</b>	<b>Comment</b>
1.0	2007-04-18		Initial Version of the Specification
1.1			
1.2	2008-02-25		
2,0	2008-06-30		Prepared for Release 2.0

## Table of Contents

1	Introduction.....	4
1.1	Overview.....	4
1.2	Scope.....	4
1.3	Symbol Definitions.....	4
1.4	Parameter Type Definitions.....	5
2	WSN Architecture.....	5
2.1	Introduction.....	5
2.2	Related Work.....	6
2.3	Proposed Architecture.....	7
3	Interfaces and Management Planes.....	9
3.1	Common Interfaces.....	9
3.2	Application -Transport Layer.....	10
3.2.1	Application Layer.....	11
3.2.2	Transport Layer.....	11
3.3	Transport - Routing Layer.....	11
3.3.1	Transport Layer .....	12
3.3.2	Routing Layer.....	12
3.4	Application - Routing Layer.....	12
3.5	Routing - Mac Layer.....	13
3.5.1	Routing Layer .....	14
3.5.2	MAC Layer .....	14
3.6	MAC - Physical Layer.....	14
3.6.1	Physical Layer .....	15
3.7	Cross Layer Management Plane (CLAMP).....	17
3.8	Energy Management Plane.....	23
3.9	Security Management Plane.....	24
3.10	Node Management.....	24
3.10.1	Delay Interfaces.....	25
3.10.2	Scheduled Interrupts Interfaces.....	26
3.11	Unified View of Interfaces and different modules.....	26
3.12	Messages.....	27
3.13	Implementation.....	27
3.13.1	PAWiS Simulation Framework.....	27
3.13.2	Resource constrained firmware.....	28

---

3.13.3 Firmware with RTOS.....28  
References.....28  
A List of Figures and Tables.....30

# 1 Introduction

## 1.1 Overview

Unlike the Internet, wireless sensor networks have yet to result in wide deployment in the real world. The unique properties of wireless communications including mobility, rapidly changing and unpredictable link quality, limited resources in terms of computation and energy, opportunistic exploitation, environmental obstructions, and new design paradigms motivates to divert from traditional layered architecture. At the same time, “plug and play” like features of the layered architecture which resulted in wide range deployment of the systems are required. In this paper we focus on the layered protocol architecture for wireless sensor networks, which provides the benefits of traditional layered architectures (interoperability) as well as focuses on cross layer design to leverage from the benefits offered by unique wireless communication properties.





## 1.2 Scope

The scope of this document is to specify a protocol architecture for Wireless Sensor Networks (WSN). The interfaces between different layers and management planes are also specified in this document.

## 1.3 Symbol Definitions

Functional Interfaces of each module may be represented as a graphical symbol as shown in Table 1.

Table 1: Description of symbols used in interface diagrams

Symbol	Description
	Function call from within a module to invoke a functional interface in an adjacent module. The user can name it anything whatever he/she wants to.
	A functional interface invoked by a callee of an adjacent layer. The name of this functional interface cannot be changed. The shaded region at the back depicts that some data is associated with its invocation. E.g. “receive” in AL is asynchronously invoked by lower layers when data arrives at the node.
	An interrupt driven functional call, used to interrupt the next upper layer on the reception of data so that the received data can be processed first. E.g. “sendAbove” in ML to interrupt the CPU so that the RL can handle the data before the CPU does something else on behalf of some other module. It can have any name.
	A functional interface called by the callee to query or notify some thing to the adjacent layer. For instance, querying regarding channel status or notifying to listen.

## 1.4 Parameter Type Definitions

Parameter types used in the document and their description can be found in Table 2. All parameters of the CLAMP database (discussed in section 3.6) are associated with a particular type. For every type an additional value “null” is introduced, which expresses an unset or unavailable value. This is the default value of all variables if not defined by the owner of the parameter.

Table 2: Parameter types used in the specifications

Parameter Type	Description
percent	8 bit unsigned integer, 0 = 0%, 255 = 100%
int8	8 bit unsigned integer
int16	16 bit unsigned integer
int24	24-bit unsigned integer
int32	32-bit unsigned integer
sint8	8-bit signed integer
enum	enumeration, values are described explicitly
time	double, time in seconds
bool	boolean, values true and false

## 2 WSN Architecture

### 2.1 Introduction

The OSI reference Model divides the network architecture into seven well defined logical layers; each layer is responsible for some specific task. In such referenced architectures, the communication between non adjacent layers is not allowed [1]. For sensor networks with constrained resources and its longevity requirements cross layer design and interaction becomes necessary. Cross layer design according to [1] is “Protocol design by the violation of reference layered communication architecture is cross-layer design with respect to the particular layered architecture.” According to [1], the violation of referenced design may include redefinition of boundaries, creation of new interfaces between adjacent and non-adjacent layers, tuning of parameters on different layers on the basis of change in network parameters from another layer, and interdependency between layers of protocol design. We define cross layer optimization as adapting certain parameters of one of the layers on the basis of feedback from another layer to achieve certain optimization (Energy efficiency or end to end delay for instance).

The authors of [2] discuss the importance of good architectural design and have emphasized that only performance enhancements at the cost of good architectural design can never result in a system which can be globally deployed like the Von Neumann, the OSI, and the Shannon’s communication architectures. The main point the authors want to stress is that, “the trade-off between performance and architecture needs to be fundamentally considered”. At the same time the traditional layered networking approach has loopholes in terms of performance and efficiency of the system [33].

We introduce a layered protocol architecture, which takes care of the issues discussed in [2], as well as provide benefits from the unique wireless communication characteristics by a cross layer approach. The proposed architecture is composed of traditional layers, including application, transport, network, link, and physical layer. The application layer (AL) (as in

Figure 1.) is in direct contact (can communicate via well defined interfaces) with routing (RL). The direct connection between AL and RL is required where no Transport Layer (TL) is used (As for most cases in sensor networks, end to end communication is not important and mostly relies on hop by hop paradigm). The interfaces between RL and Mac Layer (ML), and ML and Physical Layer (PL) are introduced. Inspired from [3], we introduce a Cross Layer management Plane (CLAMP), which we call “blackboard”, to provide cross layer benefits but in an optional way so that the concept of modularity of layered architectures is maintained. Every change in the sensor network related parameters is written to the blackboard (a shared database in the CLAMP) by the concerned layer (owner of the parameter e.g. AL is the owner of delay requirements related parameters), and any layer interested in any of the parameters can subscribe to that information, and hence it would be available to that particular layer locally with the help of a call back function. As a wireless sensor node has limited energy and it is not practical to replace the energy supply unit because of cost or geographic reasons, an energy management plane is introduced to provide services to different layers and implement key management algorithms. In most cases, security is considered as a stand-alone component of system architecture which usually is a flawed approach to network security [31]. We present a security management plane so that security can easily be integrated into every component as discussed in [31].

## 2.2 Related Work

In [4], the authors have presented a unifying link abstraction for wireless sensor networks. The main goal of [4] is to achieve generality and efficiency. They consider Sensor-net Protocol (SP) as a “narrow waist”, just like Internet protocol for the Internet. SP is an abstract layer present between the network layer and the link layer enabling different routing and MAC schemes to co-exist. They have introduced the concept of neighbor table in which data related to the neighbors are kept so that different protocols running on the same node do not keep independent routing tables and get access to the routing and link layer parameters in this shared table. Motivated by link optimizations, they have also used a message pool. Our approach is different from it in many ways. Firstly, we follow the basic architectural style as of the OSI because it is well established and successful. We do not define a message pool, neighbor table, an additional abstract layer, and additional vertical plans (other than we have defined) because of resource constraints on sensor nodes.

The authors in [5] discuss architectures for heterogeneous wireless sensor networks. They have classified the applications, routing and MAC schemes into different categories and have introduced Protocol Stack Trees (PST), which is a combination of different existing protocols and are able to satisfy different application requirements. The authors talk about cross layer entities but in a general way.

ZigBee [6] stack architecture is based on OSI reference model but considers only the layers which could achieve the required functionality for the specific market. The physical layer and the medium access control sub-layer are defined by the IEEE 802.15.4 [7] standard while ZigBee Alliance defines the layers above. Why ZigBee cannot provide a viable solution is discussed in [4] as “ZigBee proposes a classic layered architecture, but each layer assumes a specific instance of the surrounding layers: e.g., the routing layer assumes the IEEE 802.15.4 link and physical layers. An architecture build on static technologies is destined for obsolescence”.

In [8] the authors discuss network stack architecture for future sensors. They have introduced an architecture composed of Application Layer, Data Fusion Layer, Data Service Layer, Medium Access Layer and Radio Layer. The authors argue that the data fusion layer is important as it may be needed to fuse data based on application requirements or based on a fact that the sensed data may be correlated and would require data fusion. The authors further



argue that having this layer would reduce end to end latency as the message will not have to go up the stack till application layer at relay nodes or specified nodes. The data service layer essentially serves the purpose of a routing layer with functionalities like logical naming and filtering, packet gathering and scattering, and next hop determination. The MAC and routing layer serves the purpose as in traditional architectures. They have also introduced an Information Exchange Layer as a shared database that serves the purpose of cross layer optimization. In the proposed architecture, no emphasis is given to security concerns and it is stated that the data service layer can handle it. It also lacks energy management plane as energy efficiency and system life time is one of the main challenges faced by research community in this specific area.

In [Lim06], the authors propose a cross layer optimization frame with an optimization agent which provides top down and bottom up feedback to different layers of the protocol stack to benefit from the current network conditions. In principle our approach is similar to the one discussed in [Lim06], but we define a set of well-known parameters in advance which can effect network performance and energy utilization at run time. Knowing the set of parameters in advance, modules on different layers can be exchanged with any modifications in the entire protocol stack. We also keep the usage of these parameters to be optional so that if a particular module on some layer does not want to use them, the architecture should be flexible enough to accommodate this.

[Sri04] presents the benefits of cross layer feedback and related survey but does not propose a specific architecture for cross layer design.

The authors in [Rai04] present cross layer feedback architecture for wireless networks. They introduce tuning layers (to provide interfaces to data structures stored on different layers) and optimization subsystems (algorithms for cross layer optimizations) to avail cross layer benefits. As the architecture is proposed for wireless networks; the processing overhead of tuning layers and optimizations subsystems may not be well suited for low power wireless sensor networks.

## 2.3 Proposed Architecture

The main focus of the proposed solution is two-fold:

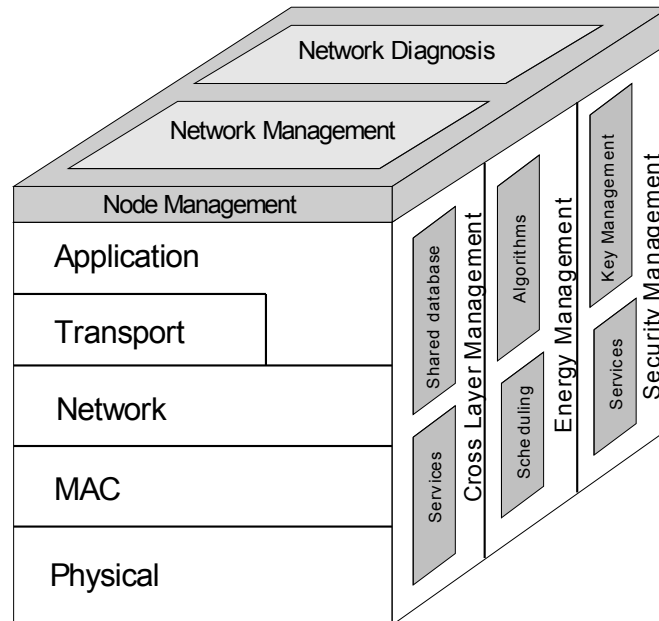
- It should be as similar as possible to the traditional layered architectures because they have already proven to be successful and have resulted in world wide deployment (e.g. TCP/IP protocol stack)
- It should deal with the unique characteristics offered by the wireless communication paradigm (e.g. mobility, rapidly changing and unpredictable link quality, limited resources in terms of computation and energy) by a cross layer design approach.

The proposed architecture comprises traditional layers and new management planes as shown in Figure 1. We include a physical, MAC, routing, an optional transport and an application layer similar to the OSI model [32]. The cross layer, energy, and security management planes all connect to the full set of layers for unlimited interaction to gain the full optimization potential. Network diagnosis and management (e.g. resetting nodes, remote firmware deployment, address assignment, querying availability of nodes) sits above all layers and planes (node management).

For each of the layers and planes we propose defined interfaces. Each of these entities can be implemented in several different ways (i.e. different MAC protocols) and then used interchangeably in conjunction with the other layers. The defined interfaces allow replacing one such entity without having to touch the others. For every pair of two such entities the interfaces of both sides are drawn in interface graphs. The symbols used to depict the

functional interfaces and calls are shown in table 1. We would discuss these layers and planes one by one in the proceeding sections. For an abstract layered architecture refer to Figure 1. For detailed architecture refer to Figure 2.

Interfaces are divided into three groups. Mandatory interfaces have to be implemented by the module with the specified parameters and functionality. Optional interfaces can be implemented or not. If they are implemented they have to fully conform to the specification. User defined interfaces can be defined and implemented according to the user's needs.



*Figure 1: Proposed protocol architecture for wireless sensor networks*

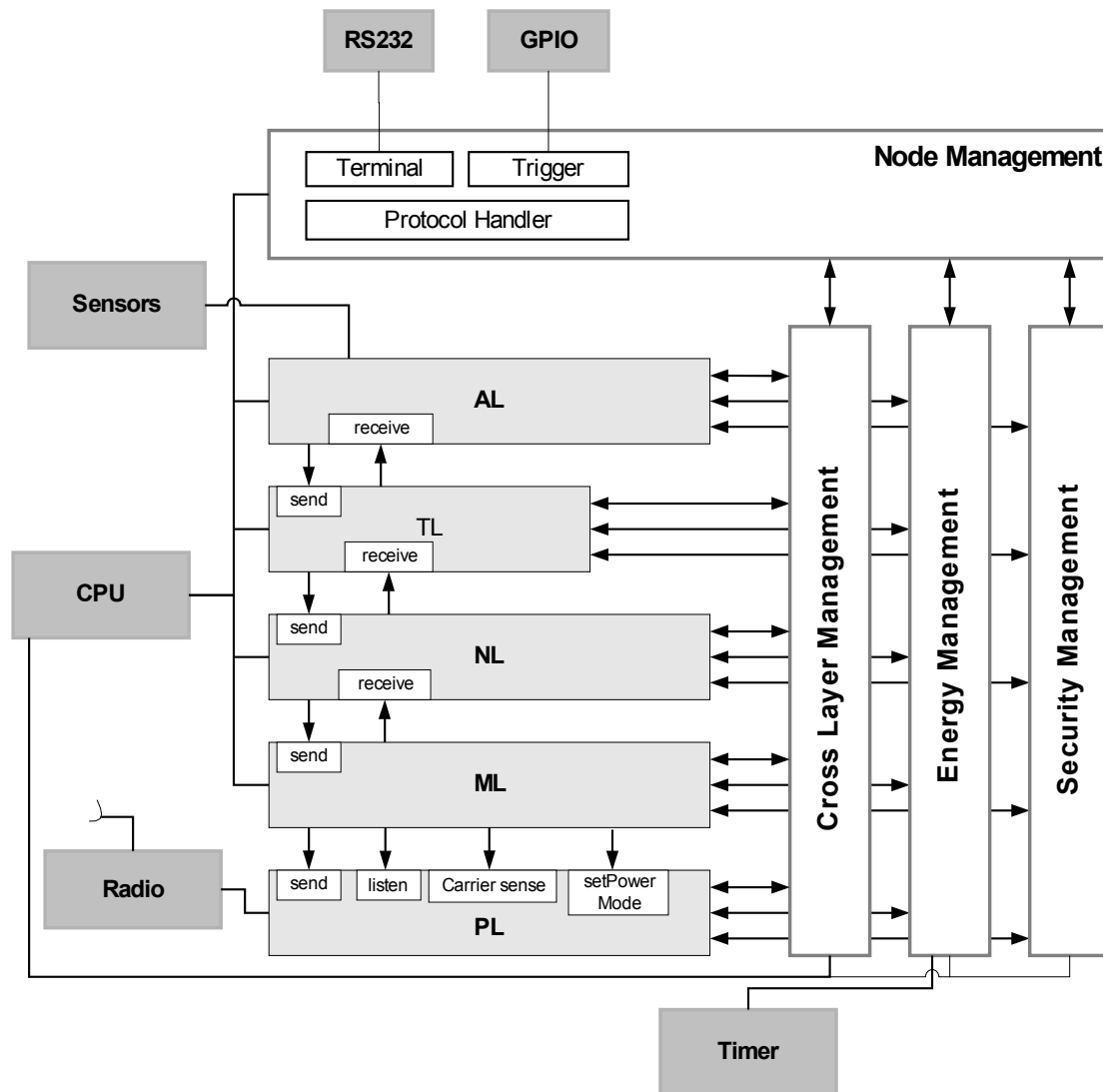


Figure 2: Proposed protocol architecture for wireless sensor networks with connections between different layers and hardware modules

## 3 Interfaces and Management Planes

### 3.1 Common Interfaces

- onChange:** It is available at all horizontal layers (AL, TL, RL, ML, PL and NM). It is discussed only once, as its syntax and semantics are the same for all layers. It is invoked by the CLAMP whenever a change occurs in a parameter to which the respective layer is subscribed.
  - In:
    - 0: (string) represents the parameter name.
    - 1: (Management::t\_ClampNotifyReason) indicates the reason. There are three possible values:

- nrNone: No reason. Inadvisable.
- NrUpdate: The value has been updated.
- NrPublish: The value has been published. This value is used when the service has subscribed to an unpublished parameter.
- 2: (variant). It contains the value of the corresponding parameter.
- **init**: It is an optional interface invoked by Nm on startup, before any other interface is invoked. Its function is to initialize the module. All the startup code must be included in this interface.
  - In:
    - 0: (int) This parameter is used to arrange different types of initialization, such as multi-part init, full init (setup+init) or reduced init (only init). The default value is 0. Other values may be implemented by users.

## 3.2 Application -Transport Layer

As the sensor nodes are very application specific, most of the sensor networks are applied to monitor a single or a group of similar phenomena. A few applications introduced previously include container tracking and monitoring [9], building automation and monitoring [10], traffic routing [11], environmental and habitat monitoring [12], health care [13], military applications [13], and smart environments [14]. In [5], the author has classified the wireless sensor networks applications on the basis of information delivery (query driven, event driven, and continuous), delay (real time, non-real time, and delay tolerant), infrastructure type (homogeneous and heterogeneous), and deployment (deterministic and non deterministic).

In [15], the author mentioned that the transport layer is required when the system has to talk to the internet or any other communication network but most of the communication within sensor networks is done hop by hop (no notion of end to end delivery in many cases), and normally there are dedicated nodes per sensor networks, which are connected to the external world.

Having discussed diverse network application requirements (combination of the above different classifications), it can be dealt with in two ways as discussed earlier: either go for an application specific architecture to attain performance gains (e.g. energy efficiency or end to end delay) at the cost of good architectural design or rely on a more generic solution at the cost of performance. As our focus is to draw the line somewhere in-between, we provide simple interfaces between AL and TL as shown in Figure 3.

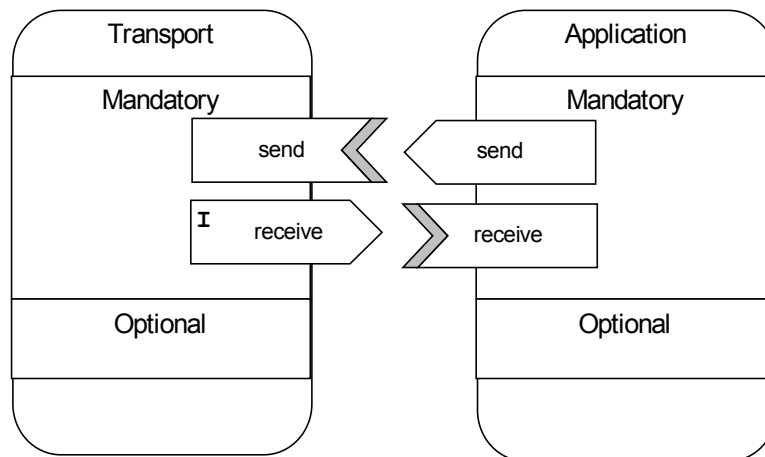


Figure 3: Interfaces between AL and TL

### 3.2.1 Application Layer

- **receive**: Mandatory interface invoked from the lower layer (transport or routing layer).
  - In:
    - 0: (AppPacket) Application data packet.
  - Out:
- 0: (int) Error value. An output of 0 or a positive value means that the operation finished without problems. A negative value means that an error occurred.

### 3.2.2 Transport Layer

- **send**: Mandatory interface invoked by the upper layer (application layer).
  - In:
    - 0: (AppPacket) Application data packet.
  - Out:
    - 0: (int) Error value. An output of 0 or a positive value means that the operation finished without problems. A negative value means that an error occurred.

## 3.3 Transport - Routing Layer

Depending on the application, the network may require a transport layer or not. Ad hoc networks do not have end-to-end communication and will not use the transport layer. However, there will be applications that do require an end-to-end communication, which is supported by the transport layer. As the transport layer is not always used, its operation must be completely transparent. The “receive” interface in the application layer works with both transport and routing layer. On the other hand, the “send” interface in the routing layer works

with both the transport and the application layer. Hence, the transport layer can be safely removed depending on the requirements.

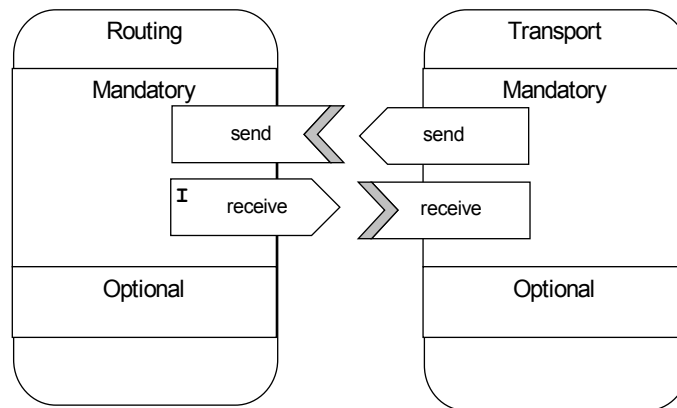


Figure 4: Interfaces between TL and RL

### 3.3.1 Transport Layer

- **receive**: Mandatory interface invoked from the lower layer (transport or routing layer).
  - In:
    - 0: (`TransportPacket`) Transport data packet.
  - Out:
    - 0: (`int`) Error value. An output of 0 or a positive value means that the operation finished without problems. A negative value means that an error occurred.

### 3.3.2 Routing Layer

- **send**: Mandatory interface invoked by the upper layer (application layer).
  - In:
    - 0: (`TransportPacket`) Transport data packet.
  - Out:
    - 0: (`int`) Error value. An output of 0 or a positive value means that the operation finished without problems. A negative value means that an error occurred.

## 3.4 Application - Routing Layer

In [16], the authors presents a survey on energy efficient routing protocols by classifying them into different categories known as data-centric, hierarchical and quality of service routing, each of them suitable for a specific application or a group of application scenarios. Mobility,

localization, and data fusion/aggregation services are also required to decrease energy utilization in wireless sensor networks. Keeping in view energy, size, and memory constraints, we provide a simple set of interfaces between RL and its adjacent layers. The rest of the components (data fusion and aggregation, localization, mobility management, forwarding, determining minimum path cost) are to be implemented within the RL as sub-modules.

We provide interfacing between AL and RL, so that if in a particular case, the TL is not implemented, the architecture is still flexible enough to accommodate this. The interfaces provided between TL and AL as well as between TL and RL are the same as between AL and RL. For interfaces between AL and RL, refer to Figure 5.

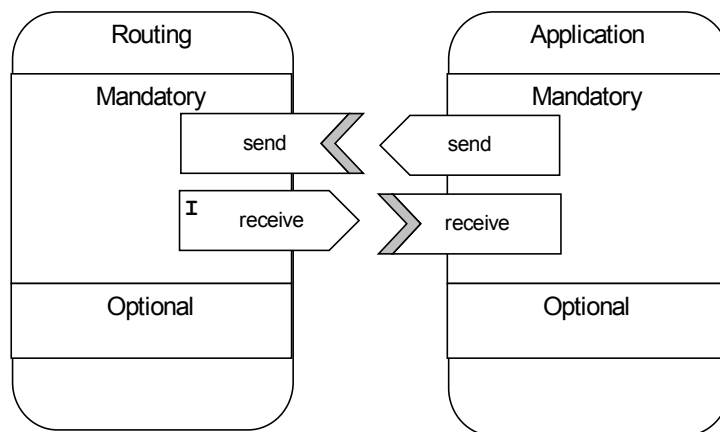


Figure 5: Interfaces between AL and RL

### 3.5 Routing - Mac Layer

The attributes of MAC schemes for wireless sensor networks include energy efficiency, scalability and adaptability to changes [17]. There is a wide range of MAC schemes ([18], [19], [20], [21], and many more) introduced previously, each of them suiting a specific group of application requirements. The interfaces between ML and RL are the same set as discussed for RL and AL (see Figure 6).

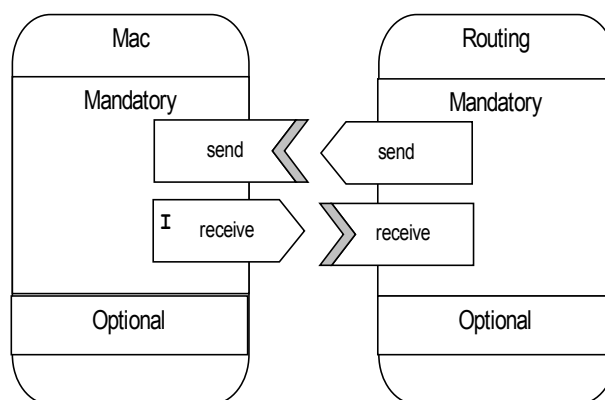


Figure 6: Interfaces between RL and ML

### 3.5.1 Routing Layer

- **receive**: Mandatory interface invoked from the lower layer (MAC Layer).
  - In:
    - 0: (`RoutingPacket`) Routing data packet.
    - 1: (`unsigned int`) The MAC address of the source.
    - 1: (`unsigned int`) The MAC address of the destination.
  - Out:
    - 0: (`int`) Error value. An output of 0 or a positive value means that the operation finished without problems. A negative value means that an error occurred.

### 3.5.2 MAC Layer

- **send**: Mandatory interface invoked from the upper layer (Routing Layer), whenever data needs to be transmitted.
  - In:
    - 0: (`RoutingPacket`) Routing data packet.
    - 1: (`unsigned int`) The MAC address of the destination.
  - Out:
    - 0: (`int`) Error value. An output of 0 or a positive value means that the operation finished without problems. A negative value means that an error occurred.

## 3.6 MAC - Physical Layer

The role of the physical layer in wireless sensor networks is not well defined yet [1]. This is because of the new modalities in wireless systems. As an example, in some radios, [22], the CRC check is implemented in hardware. Similarly, a wakeup radio [21] may require additional processing at the physical layer to figure out if the packet is intended for this specific node or not (This can be used to drop packets not intended for it and hence save processing energy at upper layers). These issues can be dealt with either by introduction of additional bits in frame headers or can be achieved with the help of user-defined interfaces available at the user's disposal. We do not provide specification for optional interfaces for this task because this cannot be uniquely solved. See Figure 7 for the interfaces between ML and PL.



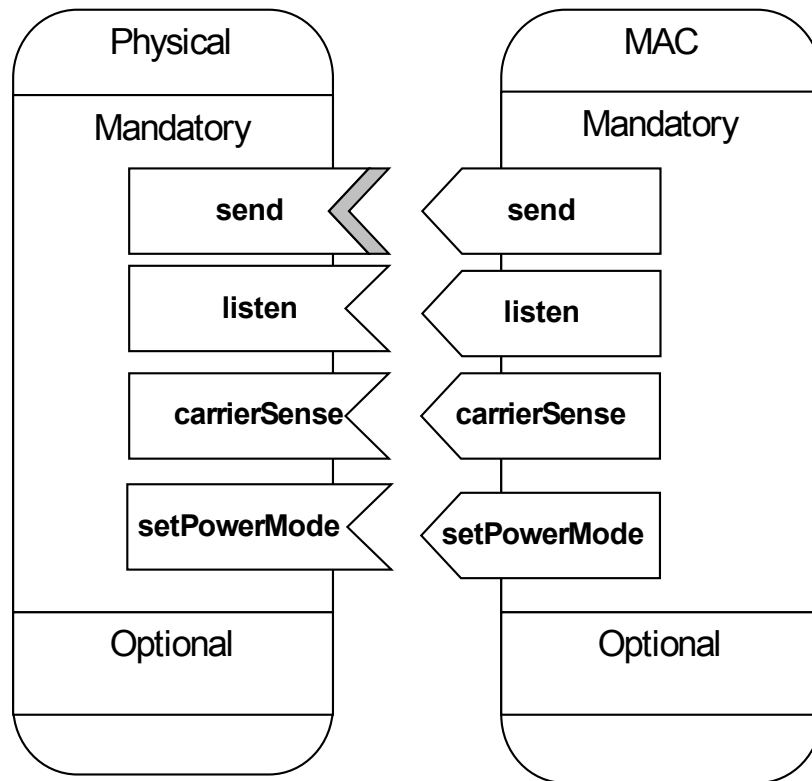


Figure 7: Interfaces between ML and PL.

### 3.6.1 Physical Layer

- **send**: Invoked by the upper layer when data needs to be submitted to the communication medium.
  - In:
    - 0: (`MacPacket`) MAC layer data packet.
- **carrierSense**: Blocking call from upper layer. Checks for a carrier signal.
  - In:
    - 0: (`simtime_t`) Time limit for carrier sensing.
  - Out:
    - 0: (`bool`) Indicates whether the carrier signal has been found.
- **setPowerMode**:
  - In:

- 0: (`int8`) Represents the power mode. 0 means off, 128 means sleep mode and 255 means full operational. The values in between the specified values can be used to utilize full granularity of different power modes. For example, the user can define further power modes, like crystal oscillator stable or PLL locked.
- **listen:**
  - In:
    - 0: (`simtime_t`) Time to listen
  - Out:
    - 0: (`PhysicalPacket`) Physical data packet. Null if none was received.
    - 1: (`int32`) Bit count of received packet. Null if none was received.
    - 2: (`int32`) Number of bit errors in the received packet. Null if none was received.
- **listenmode** (optional interface):
  - In:
    - 0: (`ListenMode_t`)
      - `lmOn`: switch Phy to permanent listen mode; this mode is only left if a packet is received or this call is called with `lmOff`.
      - `lmOff`: switch off permanent listen mode
      - `lmRetrieveData`: retrieve data of last received packet
  - Out (`In0 == lmOn`)
    - 0: (`bool`) `true`: success; `false`: error
  - Out (`In0 == lmOff`)
    - 0: (`bool`) `true`: success; `false`: error
  - Out (`In0 == lmRetrieveData`)
    - 0: (`PhysicalPacket`) Physical data packet. Null if none was received.
    - 1: (`int32`) Bit count of received packet. Null if none was received.
    - 2: (`int32`) Number of bit errors in the received packet. Null if none was received.

### 3.7 Cross LAYer Management Plane (CLAMP)

The main idea of the CLAMP is to provide a rich set of performance aware and energy aware network parameters to different layers to dynamically adapt according to application requirements. The CLAMP provides „publish”, „update”, „query”, and „subscribe” interfaces to each of the protocol stack layers (Figure 8). Initially, that CLAMP database is empty and it does not know about any of the parameters. Each layer can publish any of the parameters it owns and it wishes to share with other layers. Each layer can subscribe to parameters of interest, with the help of the „subscribe” interface. The CLAMP will „notify” these values to the subscribers if there is an „update” in the subscribed values. This is attained by the functional interface „onChange” available at each layer. The CLAMP allows only the owner of the parameter to „update” a certain value. For example, AL is the owner of parameters „delay” and „packetLoss” as shown in Figure 8. First of all, AL will „publish” these parameters to the CLAMP database. Now only AL is allowed to update these parameters, while the rest of the layers can subscribe to the desired parameters and would be notified in case of any change. If a certain layer wants to subscribe to a particular parameter which is not already published, an error message is returned. Any layer can query the particular parameter with the „query” interface if it uses the parameter rarely and does not require to be notified every time the parameter is updated. The dotted lines in Figure 8 depict the owners of different parameters in the CLAMP. We have not shown the interfaces between CLAMP and other layers other than AL because they are all the same.

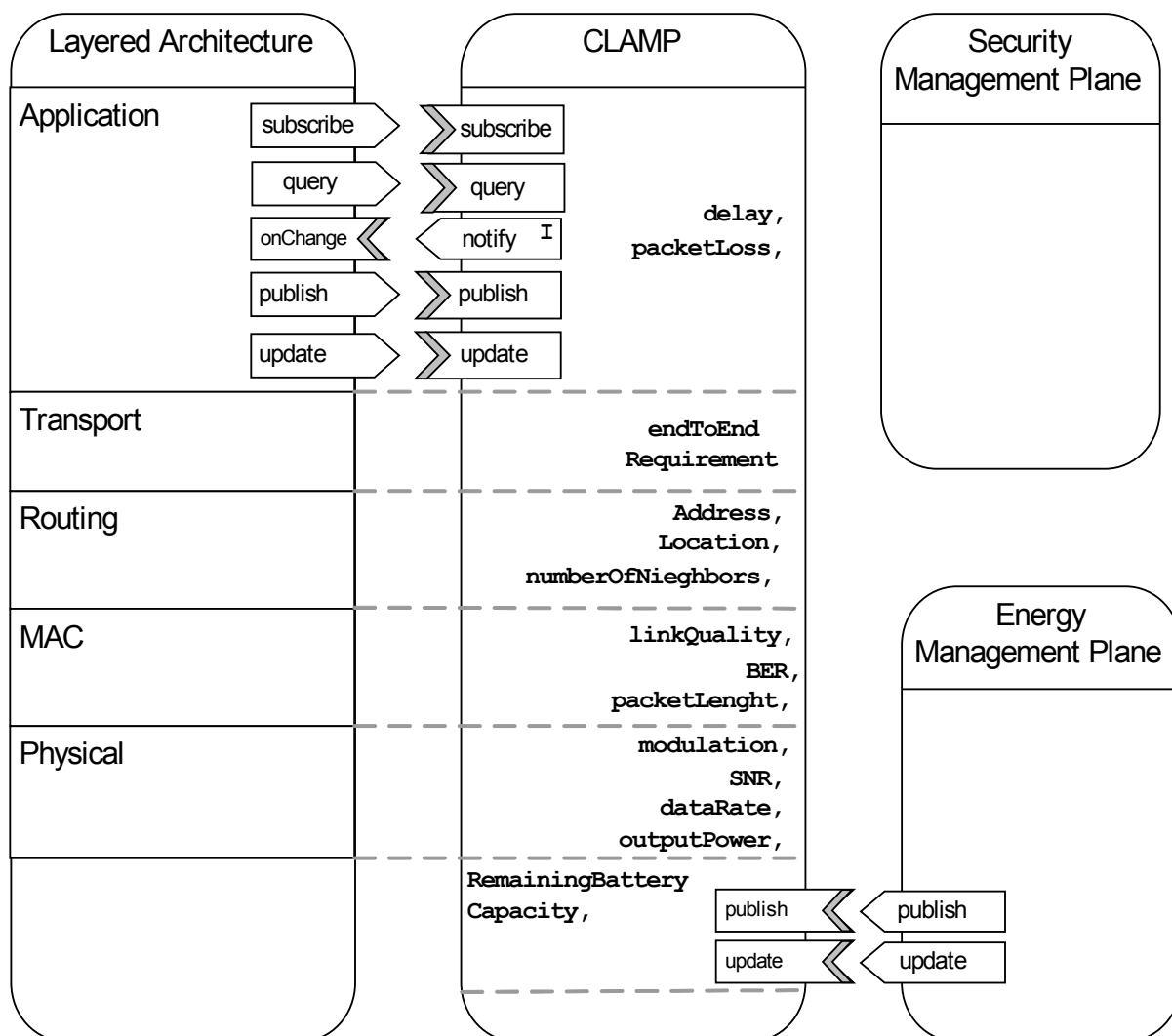


Figure 8: Interfaces between different Modules and CLAMP.

CLAMP database initially does not know about any of the parameters but these parameters are explicitly defined (though not provided to CLAMP database in the start but are well known) so that the processing overhead of parameter discovery routines may be avoided. For example, parameters “delay”, “packetLenght”, and “outputPower” (see Figure 9) are publish()ed by the respective owners and CLAMP know about these parameters. Now if ML want to subscribe() to parameter “Address”, which is not published yet, CLAMP simply would return an error message. Any of the parameter can also be query()ed by any of the modules.

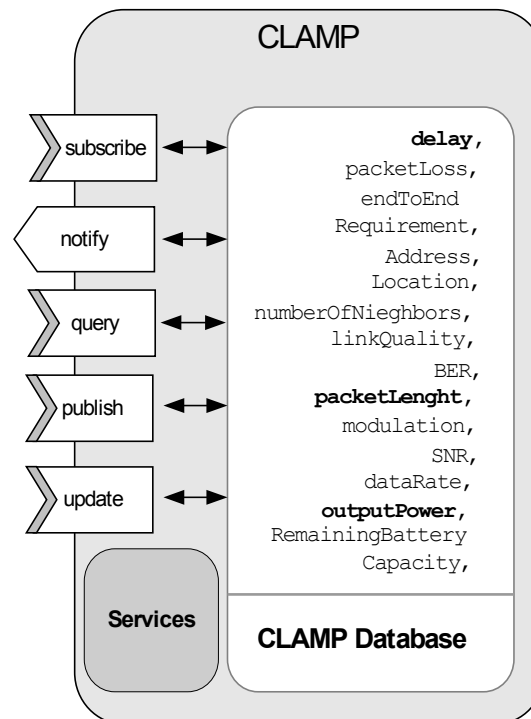


Figure 9: CLAMP architecture

- **publish:** Called by owner of the parameter to publish the parameter to the CLAMP database. A parameter with the same name must not have been already published by another module. Modules already subscribed to the parameter published get notified.
  - In:
    - 0: (string) name of the parameter being published.
  - Out:
    - 0: (bool) false on duplicate parameter name.
- **subscribe:** called by any of the layers to subscribe to a certain parameter.
  - In:
    - 0: (string) name of the parameter to subscribe.

- Out:
  - 0: (bool) true if the parameter was already published, false if it was not (yet).
- **update:** Owner of the parameter invokes this functional interface to update any of the values it owns.
  - In:
    - 0: (string) name of the parameter.
    - 1: (variant) new value of the parameter.
  - Out:
    - 0: (bool) false on error.
- **query:** called by any of the layers to query about the value of certain parameter.
  - In:
    - 0: (string) name of the parameter.
  - Out:
    - 0: (bool) status: true = ok, false = unknown parameter.
    - 1: (variant) the value of the parameter.

The network parameters that are provided by the CLAMP to different layers can be useful in many ways. See Table 2 and 3 for data types and meaning of these parameters. When and how to use these parameters is a challenging issue [1]. Table 3 also discussed potential use of each of the parameters.

Table 3: Profile and Potential Use of CLAMP Parameters

Parameter	Profile
Delay	<p>Owner: <b>Application Layer</b></p> <p>Meaning: Delay tolerance defined by the Application Layer. 0 % would mean real time application with strict end to end delay requirements. 100% means that there are absolutely no delay requirements. Note that this parameter does not state anything about the tolerable packet loss (see packetLoss below).</p> <p>Type: percent</p> <p>Potential Use: The Application Layer would define delay tolerance and Routing Layer may act accordingly, keeping in view the delay tolerance, remaining battery capacity or any other potential parameter. For least value of the delay tolerance, the routing layer may decide to sent information by minimum hop-count metric or on less congested routes to</p>

	meet the requirements of application.
packetLoss	<p>Owner: <b>Application Layer</b></p> <p>Meaning: Packet Loss Tolerance defined by Application Layer</p> <p>Type: <code>percent</code>. 0 percent means high reliability and no packet losses are acceptable while 100 % means it can tolerate high packet loss.</p> <p>Additional Considerations: A low value of packetLoss may require a Transport layer for end to end reliability and also initiates the need of acknowledged services.</p> <p>Potential Use: The Application Layer may set it to notify other layers, regarding its packet loss tolerance and the Routing Layer or Mac Layer can avoid acknowledgment messages, or retransmissions to save some energy.</p>
Address	<p>Owner: <b>Routing Layer</b></p> <p>Meaning: Logical address set by the routing layer.</p> <p>Type: <code>int16</code>; FFFFh is the broadcast address.</p> <p>Additional Considerations: The maximum numbers of nodes that can be supported are 216-1.</p> <p>Potential Use: The own address is written to this field and can be used by different layers, e.g. if a wakeup radio is used, and it is woken up by a wake-up signal including the address of the node, so it can decide what to do. The address may change from time to time depending upon Routing Layer requirements and hence included here.</p>
Location	<p>Owner: <b>Routing Layer</b></p> <p>Type: three <code>int24</code>. The location is defined by the longitude and latitude and elevation for some specific location e.g. &lt;03'37"55, 56'13"23, 837m&gt;. The full range of the <code>int24</code> is scaled to the full range of longitude and latitude, respectively. The elevation is given in meters.</p> <p>Additional Considerations: do we need date and time information to be stored as well?</p> <p>Potential Use: The reference or global location information is provided by Routing Layer and can be used by different layers, e.g. Application Layer may decide that it has already enough information regarding the required phenomenon in a specific location and it does not need this information from a group of nodes for some time (defining area dominating sets [Car05]). So this group of nodes can go to sleep to save energy. The location can also be very helpful to geographic aware routing [Ren06].</p>
noOfNeighbors	Owner: <b>Routing Layer</b>

	<p>Meaning: number of neighbors of a node.</p> <p>Type: <code>int8</code></p> <p>Additional Considerations: Theoretically, the number of neighbors can be greater than 255, but than for low power sensor networks, it would result in processing overhead in terms of maintaining the routing table and computing the lowest cost route. In case the number of neighbors is greater than 255, the first 255 neighbors with lowest route cost can be stored.</p> <p>Potential Use: This information may be utilized by the MAC layer for synchronization purposes or adaptive division of times slots for accessing the medium.</p>
linkQuality	<p>Owner: <b>MAC layer</b></p> <p>Type: <code>percent</code>; 0 = 0% means worst possible quality, 255 = 100% means excellent quality. Of course not the full granularity has to be utilized. The following values are suggested for a reduced set of states: 0..31 = bad, 32..95 = below average, 96..159 = average, 160..223 = good, 224..255 = excellent.</p> <p>Additional Considerations: The MAC layer may set only the link quality based on SNR and BER and do not provide the other parameters.</p> <p>Potential Use: If the link quality is better, the Physical can increase the data rate to exploit the opportunity or it may decrease the transit power to save energy.</p>
BER	<p>Owner: <b>MAC Layer</b></p> <p>Meaning: bit error ratio, calculated by <math>-10 \cdot \log_{10}(\text{BER})</math>, so 40 means <math>10^{-4}</math>, 70 means <math>\text{BER} = 10^{-7}</math></p> <p>Type: <code>int8</code></p> <p>Additional Considerations: A calculation model of the BER is not provided here. A receiver could estimate the BER (and/or SNR) from a received packet and its count of bit errors. In WSNs this is a major computation effort and therefore usually not provided.</p> <p>Potential Use: Depending upon BER, the Physical Layer can increase the output power or it can be compared with packet loss tolerance of the Application Layer and decide what to do.</p>
packetLenght	<p>Owner: <b>MAC layer</b></p> <p>Meaning: define packet length of a MAC packet in bytes</p> <p>Type: <code>int8</code></p>

	<p>Default Value: 0</p> <p>Additional Considerations: Here we talk about the Packet Length which is actually transmitted over the physical medium as this length is the one which affects different network variables [see table 4 for details]</p> <p>With the specified type <code>int8</code> it is only possible to transmit packets with less than 256 bytes. This current specification does not consider larger packets, MTU or segmentation and reassembly.</p> <p>Potential Use: The packet length can effect output power and bit error rate [Vij03]. Short packet sizes results in inefficient energy usage because of large overheads while long packet sizes may experience higher number of errors, so energy efficiency can be maximized by optimal packet size [Joe04].</p>
modulation	<p>Owner: <b>Physical Layer</b></p> <p>Meaning: digital modulate technique utilized for the main transceiver</p> <p>Type: <code>enum</code>: 1 = OOK, 2 = FSK, 3 = ASK, 4 = BPSK, 5 = QAM, more will be defined</p> <p>Potential Use: The modulation at Physical layer can be changed depending upon the remaining capacity of the battery [Vij03]. The number of packets in the system (in buffer or queue or being in transmission) can affect the constellation size of the modulation scheme [Abe01].</p>
SNR	<p>Owner: <b>Physical Layer</b></p> <p>Meaning: Signal-to-Noise-Ratio of the received packet expressed in dB.</p> <p>Type: <code>int8</code></p> <p>Additional Considerations: Usually a transceiver offers an RSSI value but you can't measure the noise level. Therefore the SNR value will not be provided by most implementations.</p> <p>Potential Use: If the SNR is more, and application layer has provision of delay tolerance, and the battery capacity is also low, than it can be decided to back off for some time and complete communication later on or otherwise output power can be increased.</p>
dataRate	<p>Owner: Physical Layer</p> <p>Meaning: data rate in kilo-bits per second, i.e. a value of 250 indicates a data rate of 250kbps.</p> <p>Type: <code>int16</code></p> <p>Potential Use: The lifetime of the network can be extended by using varying data rate at each node in the routing path. Reducing transmission rates at critical node (energy constrained) also results in extended network</p>



	<p>life time [Abo04]. If data rate is increased, the probability of encountering errors also increases, so a higher value of SNR would be required at the transmitting end to have an acceptable value of BER at the receiving end. Higher SNR means higher transmitting power [Wan06].Based on the data rate requirements, modulation scheme can be selected [Wan06].</p>
outputPower	<p>Owner: <b>Physical Layer</b></p> <p>Meaning: transmission power of the radio given in dBm</p> <p>Type: <code>sint8</code></p> <p>Potential Use: The modulation scheme, with certain BER threshold values and SNR can be used to calculate the transmit power [Wan06].The optimal transmit power increases with increase in the data rate (vulnerable time is decreased but thermal noise is also increased). [Fer06] A carefully chosen data rate can have high impact on transmit power and network life time [Fer06].</p>
remainingBatteryCapacity	<p>Owner: <b>Energy Management Plane</b></p> <p>Meaning: remaining battery capacity in mWs, i.e. a value of 100 means there are 100mWs. Full scale value means that remaining capacity is more than the value can represent.</p> <p>Type: <code>int16</code></p> <p>Additional Considerations: Calculating this value is difficult because the voltage characteristic (commonly used to calculate the capacity) of accumulators doesn't offer good estimations. From this value a decision should be done whether we continue to send packets or we send a "I go to sleep" packet. The resolution for such a value should fit for large batteries (10Ah) down to capacitors (1F). Therefore a value giving the relative capacity of the battery (0-100%) would be better suited.</p> <p>Potential Use: If the remaining capacity is at some threshold, than the node can back-off for some time, to allow the battery to recover and than take part in communication.</p>

### 3.8 Energy Management Plane

Sensing, communication, and processing are three main energy consuming components in a wireless sensor node [25]. As the sensor nodes are battery operated or powered by an energy scavenging technique, the restricted amount of energy in a sensor node becomes the main issue in the deployment of a sensor network. As technology advancement in the chemistry of batteries is slow compared to silicon chip technology [26], the Energy Management Plane (EMP) may provide a viable solution for efficient energy utilization.

The goal of the EMP is to maximize the network life time. When taking into consideration batteries, the actual capacities are different from rated capacities because of non linear battery

effects, different algorithms [24],[27], and [28] can be implemented in the EMP to find out the remaining capacity of the battery, which can be utilized by different layers to do energy aware computing.

The EMP may also take the responsibility for scheduling of different events to save energy. Such events include periodic listening, sensing of different types of sensors, updating timers, or analyzing incoming messages. Because it consumes time and energy to either change the state of the radio from sleep/idle to transmit state or any other hardware such as turning on the power supply of the sensors or waking up the CPU, it is very critical to implement algorithms which synchronize different activities.

For example, let's consider the radio is woken up to receive certain data, then it may prove energy efficient, to activate the sensing task and send the data to the required destination while the radio is being in wake-up state in contrast to waking up the radio again specifically to send the data.

Such energy management concepts are usually implemented implicitly in the sensor node firmware, e.g. by placing function calls in a specific order. Changing these concepts at development time is mostly tedious, whereas dynamic reconfiguration is nearly impossible. The EMP enables implementing these concepts in an explicit manner.

With this explicit approach it is easily possible to investigate various energy management concepts during development time. The defined interfaces ensure seamless interchangeability. Such an energy management concept can even dynamically change its behavior depending on the remaining energy.

The space and time complexity of these algorithms needs in-depths consideration, as we need to find out the relationship between energy utilized by these algorithms (processing energy) on individual nodes and the analysis of prolonging the network life time before implementing them.

### **3.9 Security Management Plane**

Because of limited resources, security requirements in wireless sensor networks are more challenging than in conventional networks. Security for wireless sensor networks entails key establishment and trust setup, secrecy and authentication, privacy, secure routing, intrusion detection, and secure data aggregation [29]. We provide a security management plane (SMP) similar to the “security service provider” in ZigBee Security Architecture [6] where every layer is connected to it with standardized interfaces. SMP include key management algorithm and provides security services to individual layers like helping RL in secure routing, encryption and decryption at ML, and/or authentication at TL. These functionalities are provided by a “services” component of SMP.

### **3.10 Node Management**

The node management layer deals with timing issues. There are three categories for these timing issues: delays, scheduled interrupts and periodic tasks.

### 3.10.1 Delay Interfaces

- **timerGet:** stores current timer value for later use in `timerDelayRelative()` and `timerDiff()`.
  - Out:
    - 0: (`simtime_t`) Parameter for `timerDelayRelative()` and `timerDiff()`.
- **timerDelay:** Delays the execution for a certain amount of time. The delay is stored as timer tics in an unsigned int.
  - In:
    - 0: (`simtime_t`) Delay in seconds, rounded down to timer tics.
    - 1: (`t_PowerMode`) The state the CPU and its peripherals must held during the delay.
- **timerDelayRelative:** Delays the execution for a certain amount of time relative to a previously stored timer value.
  - In:
    - 0: (`simtime_t`) Delay in seconds, rounded down to timer tics.
    - 1: (`simtime_t`) Previously stored timer value (from `getTimer()`).
    - 2: (`t_PowerMode`) The state the CPU and its peripherals must held during the delay.
- **timerDelayUntil:** Delays the execution for a certain amount of time or until a certain condition is true.
  - In:
    - 0: (`simtime_t`) Delay in seconds, rounded down to timer tics.
    - 1: (`TaskControl::Predicate*`) Determines when the delay is finished.
    - 2: (`t_PowerMode`) The state the CPU and its peripherals must held during the delay.
- **timerDelayRelativeUntil:** Delays the execution for a certain amount of time relative to a previously stored timer value or until a certain condition is true.
  - In:
    - 0: (`simtime_t`) Delay in seconds, rounded down to timer tics.
    - 1: (`simtime_t`) Previously stored timer value (from `getTimer()`).
    - 2: (`TaskControl::Predicate*`) Determines when the delay is finished.

- 3: (`t_PowerMode`) The state the CPU and its peripherals must held during the delay.
- **timerDiff**: Measures elapsed time.
  - In:
    - 0: (`simtime_t`) Previously stored timer value (from `getTimer()`).
  - Out:
    - 0: (`simtime_t`) Elapsed time in seconds.

### 3.10.2 Scheduled Interrupts Interfaces

- **timerSchedule**: Schedule interrupt after a certain amount of time.
  - In:
    - 0: (`simtime_t`) Delay in seconds, rounded down to timer tics.
- **timerScheduleRelative**: Schedule interrupt after a certain amount of time relative to a previously stored timer value.
  - In:
    - 0: (`simtime_t`) Delay in seconds, rounded down to timer tics.
    - 1: (`simtime_t`) Previously stored timer value (from `getTimer()`)
- **timerScheduleCancel**: Cancels a scheduled interrupt.

## 3.11 Unified View of Interfaces and different modules

The unified view of interfaces and all other modules is shown in Figure 10. SMP and EMP may connect to the horizontal layers of the protocol architecture via user defined interfaces.

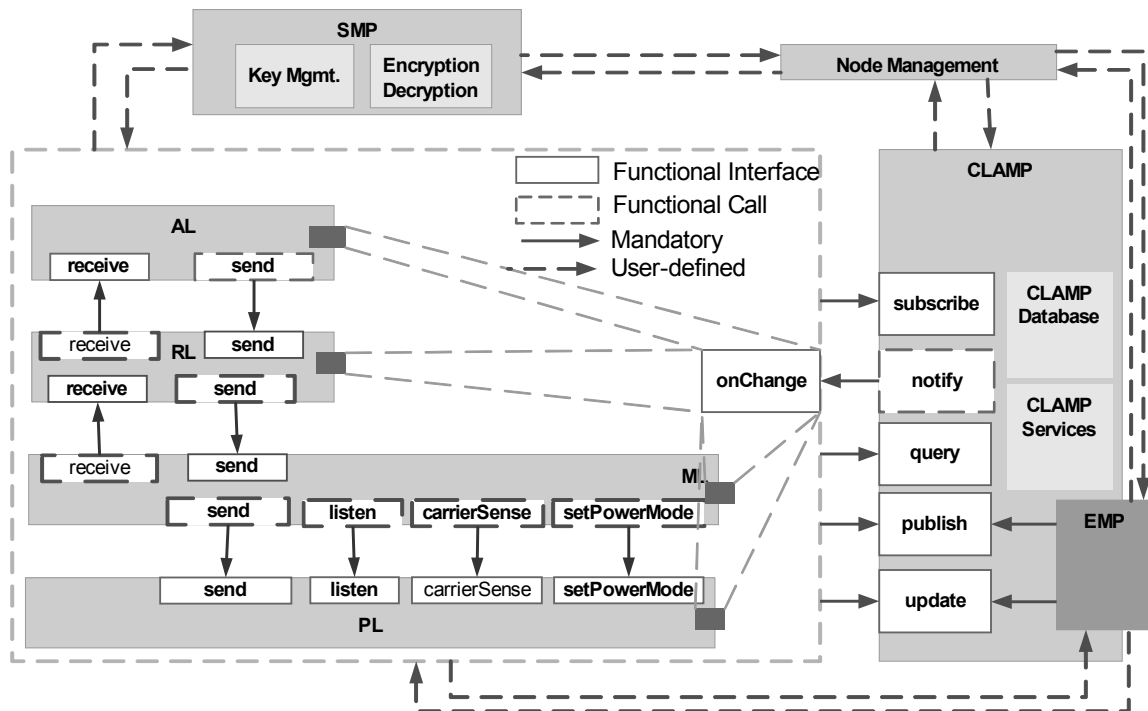


Figure 10: Unified View of Interfaces and different Modules

## 3.12 Messages

TDB

## 3.13 Implementation

### 3.13.1 PAWiS Simulation Framework

The concept can be implemented in the PAWiS framework. This uses C++ classes to model the node modules and so called functional interfaces to model interfaces (i.e. remote procedure calls) between modules. These functional interfaces are utilized to implement the communications between the network layers as well as to the CLAMP and other planes. Every invocation is managed by the discrete event simulation environment utilizing a future event list and a lot of overhead to deliver messages between the various modules.

Functional interfaces of a particular layer or plane are invoked by other modules with the PAWiS Framework method `invoke(module name, interface name, input parameters, output parameters)`. Module name is the name of the module whose interface is being called. Interface name is the name of the interface which is invoked. Third and fourth parameters within `invoke` method are pointers to object `ParameterList` used to provide input and get output respectively. For instance, assume that application layer wants to send a packet to the routing layer. It can be achieved by method `invoke("Routing", "send", &paramIn, &paramOut)`.

### 3.13.2 Resource constrained firmware

Obviously in the firmware different techniques have to be utilized for interfaces and parameters to reduce complexity and overhead. All interfaces should be implemented as regular function calls. CLAMP parameters which are only queried (i.e. no modules need immediate notification of changes) should be implemented as global variables. For CLAMP parameters with subscribed modules another approach is necessary. We propose callback function calls here.

### 3.13.3 Firmware with RTOS

TDB

## References

1. Vineet Srivastava and Mehul Motani, "Cross-Layer Design: A Survey and the Road Ahead", IEEE communication magazine, December 2005
2. Vikas Kawadia and P. R. Kumar, "A Cautionary Perspective on Cross-Layer Design", IEEE wireless communication, 2005
3. Vlado Handziski, Andreas K opke, Holger Karl, Adam Wolisz, "A Common Wireless Sensor Network Architecture", in Proc. 1. GI/ITG Fachgespräch "Sensornetze" (Technical Report TKN-03-012 of the Telecommunications Networks Group, Technische Universität Berlin), pp. 10-17, Berlin, July 2003
4. Polastre, J., Hui, J., Levis, P., Zhao, J., Culler, D., Shenker, S., and Stoica, I. 2005. A unifying link abstraction for wireless sensor networks. In *Proceedings of the 3rd international Conference on Embedded Networked Sensor Systems*, San Diego, California, USA, November 02 - 04, 2005.
5. Durrezi, A., "Architectures for heterogeneous wireless sensor networks," Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on , vol.2, no.pp. 1289- 1296 Vol. 2, 11-14 Sept. 2005
6. www.zigbee.org
7. www.ieee802.org/15/pub/TG4.html
8. Rajnish Kumar , Santashil PalChaudhuri , David Johnson, Umakishore Ramachandran, " Network Stack Architecture for Future Sensors" available as Rice Technical Report 04-447 at <http://www.cs.rice.edu/~santa/research/stack/stack.pdf>
9. S. Mahlke and S. A. Madani, "On architecture of low power wireless sensor networks for container tracking and monitoring applications", INDIN07, Vienna, Austria, 2007 [submitted]
10. M. Ulieru, S. Madani, "An Application of Industrial Agents to Concrete Bridge Monitoring," Proceedings of the 3rd International Conference on Informatics in Control, Automation and Robotics (invited), 2006.
11. Seong-Moo Yoo, "Sensor Network for Automobile Routing", a report submitted by University Transportation system of Alabama, 2004
12. A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, April 2001
13. Ning Xu, "A survey on Sensor Network Application", Computer Science Department, University of Southern California
14. Mani B. Srivastava, Richard R. Muntz, and Miodrag Potkonjak. Smart kindergarten: sensorbased wireless networks for smart developmental problem-solving environments. In *Mobile Computing and Networking*, pages 132.138, 2001
15. Akyildiz, I.F.; Weilian Su; Sankarasubramaniam, Y.; Cayirci, E., "A survey on sensor networks," Communications Magazine, IEEE , vol.40, no.8pp. 102- 114, Aug 2002
16. Kemal Akkaya and Muhammad Yousof, "A survey on Routing protocols for wireless sensor networks" retrieved on November 03, 2006 from [www.cs.umbc.edu/~kemal1/mypapers](http://www.cs.umbc.edu/~kemal1/mypapers)
17. W. Ye, J. Heidemann, D. Estrin, "Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks", *IEEE/ACM Transactions on Networking*, Volume: 12, Issue: 3, Pages:493 - 506, June 2004.
18. A. El-Hojdy and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks" in the Proceedings of the Ninth IEEE Symposium on Computers and Communication, ISCC'04, pages 244-251, Alexandria, Egypt, June 2004
19. S. Mahlke, M. Bçk, "CSMA-MPS: A Minimum Preamble Sampling MAC Protocol for Low Power Wireless Sensor Networks," Proceedings of the International Workshop on Factory Communication Systems, WFCS, pp. 73 - 80. , 2004
20. Wie Ye, John Heidemann, and Deborah Estrin, "An energy efficient mac protocol for wireless sensor networks.", In proceedings of IEEE Infocom, pp. 1567-1576, New York, NY, June 2002
21. Xue Yang, Nitin H. Vaidya, "A Wakeup Scheme for Sensor Networks: Achieving Balance between Energy Saving and End-to-end Delay," rtas, p. 19, 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04), 2004
22. [www.chipcon.com/files/220\\_Data\\_Sheet\\_1\\_2.pdf](http://www.chipcon.com/files/220_Data_Sheet_1_2.pdf)
23. Vijay T. Raisinghani, Sridhar Iyer, "Cross-layer design optimizations in wireless protocol stacks", Computer Communications 27, 2004, pp. 720-724.
24. Rakhmatov, D.; Vrudhula, S.; Wallach, D.A., "A model for battery lifetime analysis for organizing applications on a pocket computer," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.11, no.6pp. 1019- 1030, Dec. 2003
25. Du, X.; Lin, F., "Efficient energy management protocol for target tracking sensor networks," Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on , vol., no.pp. 45- 58, 15-19 May 2005
26. Gao, Q., Blow, K. Holding, D. Marshall, Ian, "Analysis of Energy Conservation in Sensor Networks", Wireless Networks, Volume 11, Number 6, pp. 787-794(8), November 2005
27. M. Handy and D. Timmermann, "Simulation of Mobile Wireless Networks with Accurate Modelling of Non-linear Battery Effects", In proceedings of Applied simulation and modeling (ASM), Marbella, Spain, 2003
28. C.F. Chiasserini, R.R. Rao, "Energy efficient battery management," Proc. of Infoconi 2000, Tel Aviv, Israel, March 2000.
29. Perrig, A., Stankovic, J., and Wagner, D. 2004. Security in wireless sensor networks. *Commun. ACM* 47, 6 Jun. 2004

30. [www.ict.tuwien.ac.at/pawis](http://www.ict.tuwien.ac.at/pawis)
31. Perrig, A., Stankovic, J., and Wagner, D. 2004. Security in wireless sensor networks. *Commun. ACM* 47, 6 (Jun. 2004), pp. 53-57.
32. ISO/IEC Standard 7498-1:1994. Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.
33. van Hoesel, L.; Nieberg, T.; Jian Wu; Havinga, P.J.M. “Prolonging the lifetime of wireless sensor networks by cross-layer interaction”, *Wireless Communications, IEEE* [see also *IEEE Personal Communications*], Vol.11, Iss.6, Dec. 2004, Pages: 78- 86.
- [Lim06] Su, W. and Lim, T. L. 2006. Cross-Layer Design and Optimization for Wireless Sensor Networks. In Proceedings of the Seventh ACIS international Conference on Software Engineering, Artificial intelligence, Networking, and Parallel/Distributed Computing (Snpd'06) - Volume 00 (June 19 - 20, 2006). SNPD-SAWN. IEEE Computer Society, Washington, DC, 278-284
- [Sri04] Vijay T. Raisinghani, Sridhar Iyer, “Cross-layer design optimizations in wireless protocol stacks”, *Computer Communications* 27, 2004, pp. 720-724.
- [Rai06] Raisinghani, V.T.; Iyer, S. “Cross-layer feedback architecture for mobile device protocol stacks”, *Communications Magazine, IEEE*, Vol.44, Iss.1, Jan. 2006 Pages: 85- 92
- [Ren06] Zeng, K., Ren, K., Lou, W., and Moran, P. J. 2006. Energy-aware geographic routing in lossy wireless sensor networks with environmental energy supply. In Proceedings of the 3rd international Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (Waterloo, Ontario, Canada, August 07 - 09, 2006).
- [Abo04] AbouGhazaleh, N.; Lanigan, P.; Gobriel, S.; Mosse, D.; Melhem, R. “Dynamic rate-selection for extending the lifetime of energy-constrained networks”, *Performance, Computing, and Communications, 2004 IEEE International Conference on*, Vol., Iss., 2004 Pages: 553- 558.
- [Wan06] Wang, W., Peng, D., Wang, H., and Sharif, H. 2006. Study of an energy efficient multi rate scheme for wireless sensor network MAC protocol. In Proceedings of the 2nd ACM international Workshop on Quality of Service & Security For Wireless and Mobile Networks (Terromolinos, Spain, October 02 - 02, 2006).
- [Abe01] Schurgers, C.; Aberthorne, O.; Srivastava, M.B., “Modulation scaling for energy aware communication systems”, *Low Power Electronics and Design, International Symposium on*, 2001., Vol., Iss., 2001,Pages:96-99.
- [Joe06] Inwhae Joe, “Optimal packet length with energy efficiency for wireless sensor networks”, *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, Vol., Iss., 23-26 May 2005 Pages: 2955- 2957 Vol. 3.
- [Fer06] Panichpapiboon, S.; Ferrari, G.; Tonguz, O.K. “Optimal Transmit Power in Wireless Sensor Networks”, *Mobile Computing, IEEE Transactions on*, Vol.5, Iss.10, Oct. 2006 Pages: 1432- 1447.
- [Car05] Carreras, I.; Chlamtac, I.; Woesner, H.; Zhang, H., “Nomadic sensor networks”, *Proceedings of the Second European Workshop on Wireless Sensor Networks*, Jan.-2 Feb. 2005 Pages: 166- 175.

## A List of Figures and Tables

Figure 1: Proposed protocol architecture for wireless sensor networks.....	8
Figure 2: Proposed protocol architecture for wireless sensor networks with connections between different layers and hardware modules.....	9
Figure 3: Interfaces between AL and TL.....	11
4.....	13
Figure 5: Interfaces between TL and RL.....	13
Figure 6: Interfaces between AL and RL.....	15
Figure 7: Interfaces between RL and ML.....	16
Figure 8: Interfaces between ML and PL.....	17
Figure 9: Interfaces between different Modules and CLAMP.....	19
Figure 10: CLAMP architecture.....	20
Figure 11: Unified View of Interfaces and different Modules.....	27
Table 1: Description of symbols used in interface diagrams.....	4
Table 2: Parameter types used in the specifications.....	5
Table 3: Profile and Potential Use of CLAMP Parameters.....	21